IN THE CLAIMS:

Please amend claims 1, 4, 5, 8, 9, 11 and 15, and cancel claims 2, 3, 6, 14 and 18 as follows:

1. (currently amended) A method for processing tasks in a data processing system including a microprocessor and an instruction cache wherein tasks of various types are defined in the system, each task type having code associated therewith, the tasks being processed ~~out of sequential order~~ by loading the associated code into the instruction cache for execution on the microprocessor, the method comprising: ~~the step of~~

placing tasks of same task type into a batch such that the tasks in a batch are processed before processing ~~the~~ a next ordered task; ~~and~~

if the code associated with at least one type of task fits within the instruction cache, processing such a task by loading the associated code into the instruction cache and executing the code on the microprocessor, and, on a determination that there is a further task of like type in the batch, executing the loaded code to process the further task;

if the code associated with at least one type of task is not capable of being loaded as a whole into the instruction cache, logically dividing the code at one or more break points into two or more portions, and wherein during processing of such a task, responding to a break point defined within a first portion of the code to schedule a further task for future execution of a second portion of the code;

wherein the tasks of the same task type use same code in the instruction cache; and

wherein the tasks are placed in the batch at the time the tasks are scheduled.

2. (cancelled)

3. (cancelled)

4. (currently amended) A method as claimed in claim ~~3~~ 1 wherein the further scheduled task is placed in a batch of like tasks.

5. (currently amended) A method as claimed in claim ~~3~~ 1 wherein each of portions of code defines an atomic operation.

6. (cancelled)

7. (previously presented) A method as claimed in claim 1 wherein the tasks are managed as a queue.

8. (currently amended) A computer program product comprising a computer ~~usable~~ readable medium having computer readable program code ~~means~~ embodied in the medium for processing tasks in a data processing system, the data processing system including a microprocessor and an instruction cache and wherein tasks of various types are defined in the system, each task type having code associated therewith, the tasks being processed ~~out of sequential order~~ by executing the associated code on the microprocessor, the program code ~~means~~ comprising code ~~means~~ for:

scheduling tasks of same type into a batch such that tasks in a batch are processed before processing ~~the~~ a next ordered task, wherein the tasks of the same type use the same program code;

if the code associated with at least one type of task fits within the instruction cache, processing such a task by loading the associated code into the instruction cache and executing the code on the microprocessor, and, on a determination that there is a further task of like type in the batch, executing the loaded code to process the further task;

if the code associated with at least one type of task is not capable of being loaded as a whole into the instruction cache, logically dividing the code at one or more break points into two or more portions, and wherein during processing of such a task, responding to a break point defined within a first portion of the code to schedule a further task for future execution of a second portion of the code;

wherein the tasks of the same task type use same code in the instruction cache; and

wherein the tasks are placed in the batch at the time the tasks are scheduled.

9. (currently amended) Data processing ~~apparatus~~ system comprising a microprocessor and an instruction cache wherein tasks of various types are defined in the system, each task type having code associated therewith, the ~~apparatus including~~ system comprising:

means for processing the tasks ~~out-of-sequential-order~~ by loading the associated code into the instruction cache for execution on the microprocessor; and

if the code associated with at least one type of task fits within the instruction cache, means for processing such a task by loading the associated code into the instruction cache and executing the code on the microprocessor, and, on a determination that there is a further task of like type in the batch, executing the loaded code to process the further task;

if the code associated with at least one type of task is not capable of being loaded as a whole into the instruction cache, means for logically dividing the code at one or more break points into two or more portions, and wherein during processing of such a task, responding to a break point defined within a first portion of the code to schedule a further task for future execution of a second portion of the code;

means for scheduling tasks of same type into a batch, wherein the means for processing the tasks is operable to process the tasks in a batch before processing ~~the~~ a next ordered task; and

wherein the tasks of the same type use the same program code; and

wherein the tasks are placed in the batch at the time the tasks are scheduled.

10. (original) Data processing apparatus as claimed in claim 9 wherein the microprocessor and ~~i-cache~~ instruction cache are embodied on a single chip.

11. (currently amended) A computerized method for scheduling tasks in a task queue, the method comprising:

identifying a new task to be scheduled in the task queue;

determining if the task queue includes a cached task that requires ~~the~~ a same code to process the cached task and the new task; and

batching the new task with the cached task if the task queue includes the cached task that requires the same code to process the cached

task and the new task;

if the code associated with at least one type of task fits within the instruction cache, processing such a task by loading the associated code into the instruction cache and executing the code on the microprocessor, and, on a determination that there is a further task of like type in the batch, executing the loaded code to process the further task;

if the code associated with at least one type of task is not capable of being loaded as a whole into the instruction cache, logically dividing the code at one or more break points into two or more portions, and wherein during processing of such a task, responding to a break point defined within a first portion of the code to schedule a further task for future execution of a second portion of the code; and

wherein the tasks are placed in the batch at the time the tasks are scheduled.

12. (previously presented) The method of claim 11, further comprising adding the new task to the end of the queue if the task queue does not include the cached task that requires the same code to process the cached task as the new task.

13. (previously presented) The method of claim 11, further comprising:

loading task code for processing the cached task into an instruction cache;

executing the task code for processing the cached task in the instruction cache; and

executing the task code for processing the new task in the instruction cache without loading new code into the instruction cache.

14. (cancelled)

15. (currently amended) A computer program product embodied in a tangible computer readable media for processing tasks in a data processing system comprising:

~~computer-readable~~ program codes ~~coupled to the tangible media~~ for scheduling tasks in a task queue, the ~~computer-readable~~ program codes

comprising codes ~~configured to cause the program~~ to:

identify a new task to be scheduled in the task queue;

determine if the task queue includes a cached task that requires the same code to process the cached task and the new task; ~~and~~

batch the new task with the cached task if the task queue includes the cached task that requires the same code to process the cached task and the new task;

if the code associated with at least one type of task fits within the instruction cache, process such a task by loading the associated code into the instruction cache and execute the code on the microprocessor, and, on a determination that there is a further task of like type in the batch, execute the loaded code to process the further task;

if the code associated with at least one type of task is not capable of being loaded as a whole into the instruction cache, logically dividing the code at one or more break points into two or more portions, and wherein during processing of such a task, respond to a break point defined within a first portion of the code to schedule a further task for future execution of a second portion of the code; and

wherein the tasks are placed in the batch at the time the tasks are scheduled.

16. (previously presented) The computer program product of claim 15, wherein the program code is further configured ~~to cause the program~~ to add the new task to the end of the queue if the task queue does not include the cached task that requires the same code to process the cached task as the new task.

17. (previously presented) The computer program product of claim 15, wherein the program code is further configured ~~to cause the program~~ to:

load task code for processing the cached task into an instruction cache;

execute the task code for processing the cached task in the instruction cache; and

execute the task code for processing the new task in the instruction cache without loading new code into the instruction cache.

18. (cancelled)